# CSE 531 Homework Assignment 2

Due in class on Tuesday, Sep 25.

September 9, 2007

There are totally 5 problems, 10 points each. You should do them all. We will grade only 4 problems chosen at my discretion. If it so happens that you don't do one of the problem we don't grade, then no points will be deducted.

**Problem 1.** A climatologist has a large data set of temperatures recorded daily for more than a century. To study global warming trend, he would like to find a period during which the daily average temperature was increased the most. Specifically, he has an array of average temperatures $\mathbf{t} = [t_1, t_2, \ldots, t_n]$, where $t_i$ is the average temperature of the $i$th day on record. He would like to find a pair of day $(i, j)$ for which $i < j$ and $t_j - t_i$ is the largest among all such pairs. Help him design an $O(n \lg n)$-time divide-and-conquer algorithm to find such a pair.

**Problem 2.** An FTP server has a large batch of file downloading requests to process. There are $n$ requests in the batch. For $1 \le i \le n$, the $i$th request came at time $t_i$ asking for a file of size $f_i$ bytes. The $t_i$ are distinct and are not sorted in any order. The total number of requested bytes is $F = \sum_{i=1}^{n} f_i$. Unfortunately, the server can only serve $F/2$ bytes at this moment.

The server wants to serve the maximum amount of data, while honors requests which come first. Hence, the server needs to find a request $k$, for some $1 \le k \le n$, such that the total number of bytes requested *before* $t_k$ is no more than $F/2$, and that the total number of bytes requested *after* $t_k$ is strictly less than $F/2$. (We do not count request $k$ as before or after $t_k$.)

(a) Show that such a request $k$ always exists.

(b) Devise a $O(n)$-time algorithm to find such a request $k$.

(After $t_k$ is found, a simple scan through all requests will pick out requests to be processed. **Hint:** LINEAR-SELECT might help.)

**Problem 3.** Let $\mathbf{t} = [t_1, \ldots, t_{2n-1}]$ be any array of $2n-1$ real numbers. For each such $\mathbf{t}$, define an $n \times n$ matrix $\mathbf{A_t} = (a_{ij})_{1 \le i,j \le n}$ by setting $a_{ij} = t_{i-j+n}$. This type of matrices arise quite naturally in the study of discrete time random processes, when we often have to perform matrix operations involving these matrices.

(a) Let $\mathbf{v}$ be any column vector with $n$ coordinates, i.e. $\mathbf{v} \in \mathbb{R}^n$. Show how to compute the product $\mathbf{A_t v}$ in time $O(n \lg n)$ for any $\mathbf{t}$ and $\mathbf{v}$.

(b) Let $\mathbf{s}, \mathbf{t}$ be any two arrays of $2n-1$ real numbers. Show how to compute the product $\mathbf{A_s A_t}$ in time $O(n^2)$.

**Problem 4.** We are given $n$ intervals on the real line, i.e. $[a_i, b_i]$, $a_i \le b_i$, for $1 \le i \le n$. Each interval overlaps at least $d-1$ other intervals, for some $1 \le d \le n$. Moreover, no interval contains another.

You want to *crazy-sort* these intervals in the following sense: you are to produce a permutation $(i_1, i_2, \ldots, i_n)$ of $\{1, \ldots, n\}$ such that, for all $j \in \{1, \ldots, n\}$ there exists a $c_j \in [a_{i_j}, b_{i_j}]$ satisfying $c_1 \le c_2 \le \cdots \le c_n$. Devise an $O(n \lg \frac{n}{d})$-time algorithm to crazy-sort these intervals.

**Problem 5.** The Buffalo Tennis Association is about to establish a new Tennis tournament called the BUFOPEN. The organizers do not agree with the current "binary tree" format of the Grand Slams. They think it is unfair to the second best player as he/she could have been beaten in rounds before the final match by the champion, and thus does not have a chance to be the runner-up. (Pretty much everyone belonging to Federer's branch is out of luck. Think Rodick!)

Here we assume the players' strengths are totally ordered, namely if player $A$ beats player $B$, and $B$ beats $C$, then we can safely conclude that $A$ is stronger than $C$ without playing $C$.

Consequently, the organizers of the BUFOPEN ask the students of CSE 531 to design a tournament which can decide the best and the second best players in as few matches as possible.

1. Design such a tournament which uses only $n + \lceil \lg n \rceil - 2$ matches, where $n \geq 2$ is the total number of players.

2. If we also want to determine the third best player, in addition to the first and the second best, how many matches do you need in total?

3. Suppose there is a tournament format which determines the $m$th best player $X$, where $1 < m \leq n$. Show that after the tournament, one can identify the set of $m-1$ players who are stronger than $X$ without any additional matches.

   (**Hint:** Let $\mathcal{S}$ be the set of players who have beaten $X$ directly or indirectly, then all players in $\mathcal{S}$ are stronger than $X$. Prove that if $|\mathcal{S}| < m - 1$, then there is some other ordering of players' strengths for which $X$ is no longer the $m$th strongest, but the matches' results are exactly the same as before.)